Vitor Amadeu Souza

Desenvolvimento de Projetos com

ARM

programado em C no

CubeIDE

Com base no modelo STM32F103C8
Volume único

© 2020 by Cerne Tecnologia e Treinamento Ltda.

© 2020 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Setembro de 2020

Direitos reservados por:
Cerne Tecnologia e Treinamento Ltda
Produção: Cerne Tecnologia e Treinamento
E-mail da Empresa: cerne@cerne-tec.com.br
Home Page: www.cerne-tec.com.br.com.br
Atendimento ao Consumidor: sac@cerne-tec.com.br
Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

Dedicatória

Como nos meus outros livros, dedico este livro a minha querida esposa Renata Leal.

"É claro que assim como o nosso caminhar não passa de uma queda constantemente evitada, também nossa vida não é mais que uma morte constantemente adiada."

Arthur Schopenhauer

Kits Didáticos e Gravadores da Cerne Tecnologia

A Cerne tecnologia têm uma linha completa de aprendizado para os ARMes da família PIC, 8051, Holtek, dsPIC, ARM, ARM, etc. Veja os detalhes de um de nossos kits.



Kit Cerne ARM

- Microcontrolador ATMEGA8;
- Comunicação serial RS232;
- Alimentação de 12V;
- Pinos de I/O;
- Gravação ICSP.

Uma linha completa de componentes para o desenvolvimento de seus projetos eletrônicos como displays, PICs, botões, leds, cristais e etc. Visite a nossa página na Internet, no endereço **www.cerne-tec.com.br** e conheça melhor nossos serviços e produtos.

Cerne
Conhecimento para o Desenvolvimento

www.cerne-tec.com.br

Sumário

01. Metodologia de desenvolvimento **6**

02. Amperímetro **7**

03. Motor de passo bipolar **21**

04. VU Meter **50**

05. Horímetro **60**

06. SGA **70**

07. Dinamômetro **84**

08. Expansão de saídas com shift register **97**

09. Expandindo o número de entradas com shift register **118**

10. Varredura de displays de 7 segmentos **135**

11. Medindo a concentração de GLP **146**

12. Medindo a concentração de H_2 **163**

13. Medindo a concentração de Amônia **179**

14. Medindo a concentração de Benzeno **197**

15. Célula de Carga **214**

16. Decibelímetro **229**

17. Medindo o fluxo de massa **248**

18. Medindo a concentração de Ozônio **263**

19. Medição de pH **280**

21. Medição de temperatura com 1N4148 **313**

22. Projetos com Energia Solar **326**

23. Gaussímetro **343**

24. Medindo Álcool em ppm **360**

25. Medição de Monóxido de Carbono **374**

26. Medição de Dióxido de Carbono **394**

27. Medição de pressão **413**

28. Medição de umidade **424**

29. Medição de corrente de 4 a 20 mA **435**

30. Medição de temperatura com NTC **446**

31. Medição de tensão RMS **457**

32. Medição de temperatura do Pt100 **469**

Capítulo 1
Metodologia de desenvolvimento

1. Introdução

Esta literatura é uma continuação da obra *Programando o ARM em C no CubeIDE - Com base no modelo STM32F103C8* (2020) do mesmo autor e editora, onde outros exemplos são explorados de modo que o leitor possa ampliar seu embasamento teórico e prático para desenvolver mais aplicações nesta ferramenta.

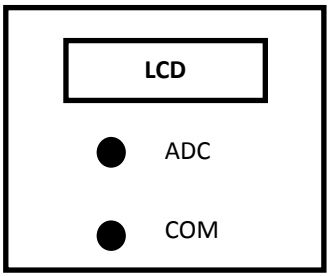
É importante que o leitor tenha ciência dos assuntos abordados na obra citada, para que haja um melhor aproveitamento do conteúdo a ser apresentado.

A placa didática utilizada é a STM32F103C8T6 *Blue Pill* além de um gravador modelo ST-LINK v2, onde os recursos para testar os circuitos propostos são conectados através das conexões disponíveis na lateral do kit. Tal kit e gravador estão à venda no site www.cerne-tec.com.br.

Capítulo 2
Amperímetro

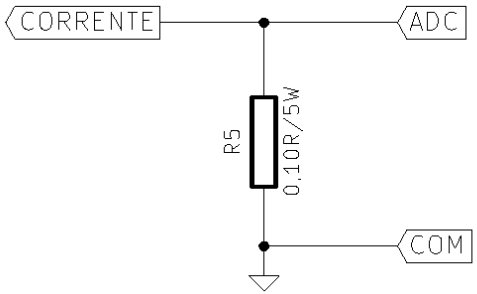
1. Amperímetro

Um amperímetro é um instrumento capaz de medir a corrente de um circuito e apresentá-la em uma interface homem-máquina, que pode ser do tipo display de 7 segmentos, LCD, display gráfico, etc. Estes instrumentos possuem uma baixa-impedância de entrada, para que assim não interfiram no funcionamento do circuito mensurado e podem medir ampla faixa de corrente a nível de corrente contínua (DC) ou alternada (AC). Neste capítulo, será desenvolvido um amperímetro DC utilizando uma entrada analógica do ARM, onde o resultado mensurado é apresentado em um display LCD, conforme sugere o modelo apresentado a seguir.



Como o amperímetro fica em série com o circuito a ser mensurado, a entrada COM representa a entrada enquanto que ADC representará a saída do medidor.

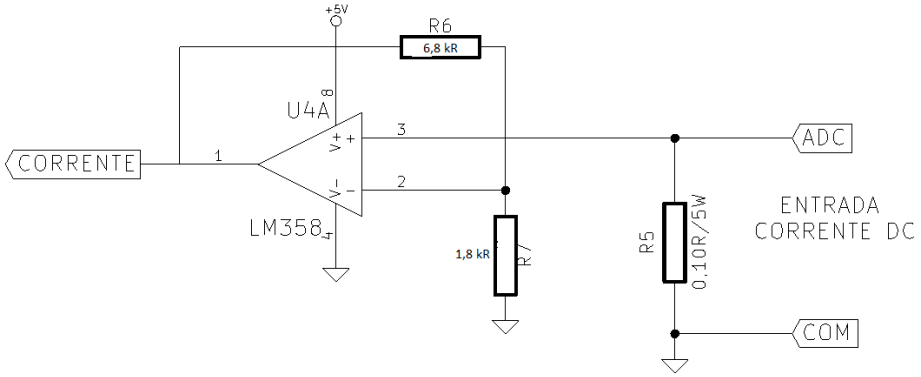
A faixa de corrente contínua projetada para medição será de 0,0 A a 7,0 A. Como o ARM mede na faixa de 0 V a 3,3 V, é necessário um circuito de conversão corrente para tensão utilizando um resistor shunt de 0,10Ω/5W, que fornece uma tensão proporcional a corrente que passa pelo mesmo. Observe o circuito de conversão corrente-tensão a seguir que oferece o funcionamento comentado.



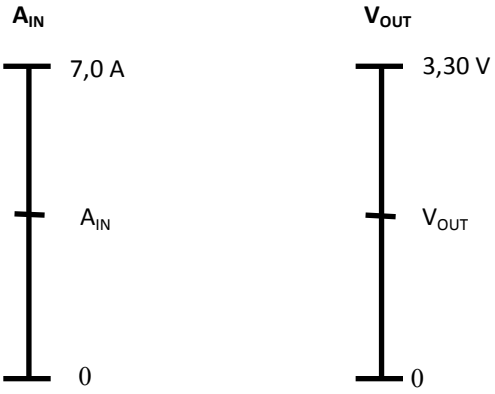
As entradas ADC e COM servem para realizar a medição no circuito enquanto que a saída CORRENTE fornece a tensão proporcional a corrente que passa pelo resistor shunt. Quando a corrente de 7 A passar pelo circuito, é observado a seguinte tensão no terminal CORRENTE, segundo a 1º Lei de Ohm.

$$V = R.I$$
$$V = 0{,}10.7$$
$$V = 0{,}70V$$

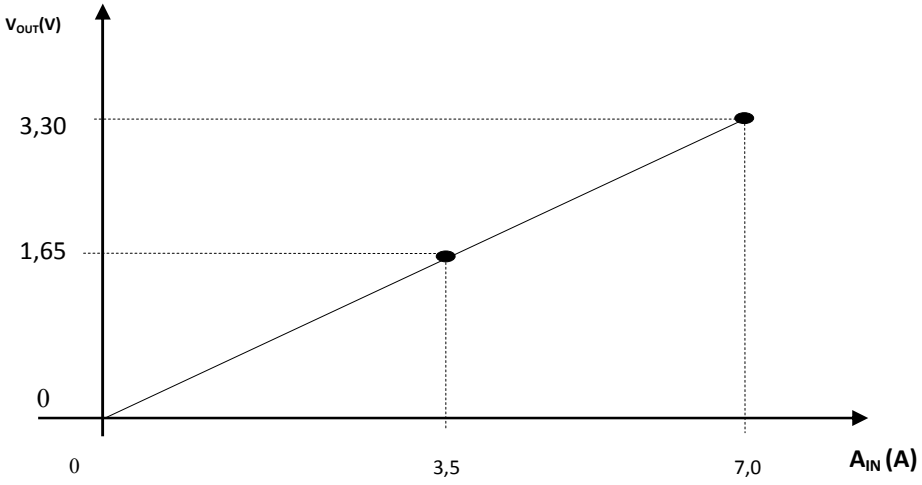
Esta tensão é muito pequena para ser tratada pelo conversor ADC do ARM que possui faixa de 0 a 3,3 V. Sendo assim é necessário amplificá-la através de um amplificador operacional (AOP) não inversor com ganho de 4,7. Isso é obtido através do circuito abaixo.



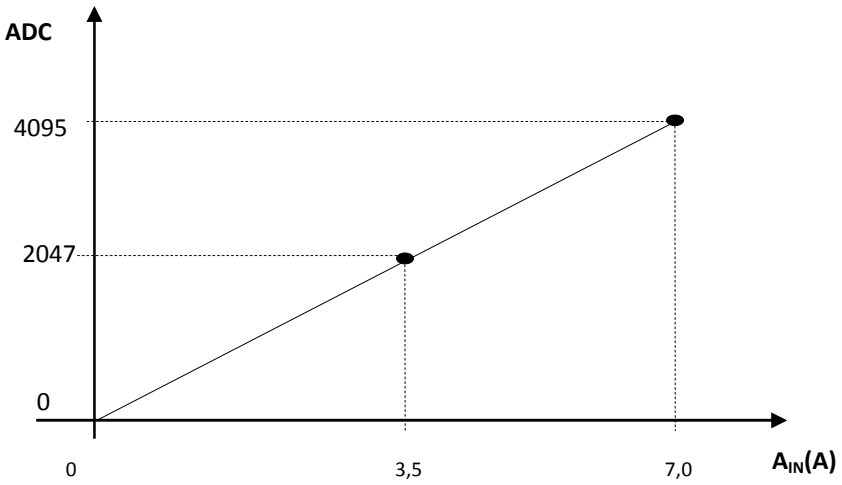
A tensão de 3,3V será a máxima na entrada analógica do ARM. Desta forma, a figura abaixo mostra a relação da corrente medida e da entrada analógica do ARM.



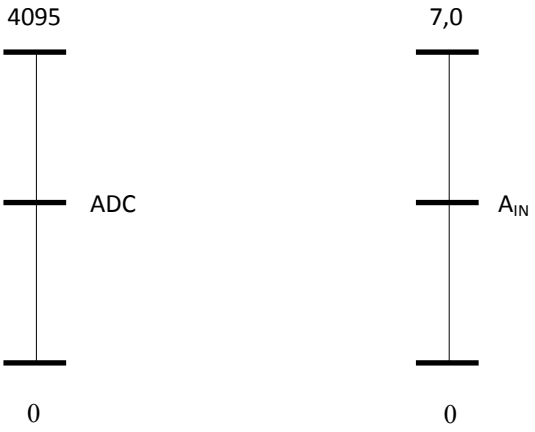
A curva que relaciona a tensão de saída (V_{OUT}) com a corrente de entrada (A_{IN}) tem a característica apresentada a seguir.



O conversor ADC do ARM é de 12 bits, o que indica que o mesmo é capaz de ter uma resolução de 4096 pontos, já que 2^{12} é igual a este valor. A tensão a ser medida está ligada a uma das entradas analógicas do ARM, no qual fornece como resultado um valor binário na faixa de 0 a 4095 de acordo com a tensão de 0 a 3,3 V. Sendo assim, pode-se modelar o seguinte gráfico, no qual converte a escala de entrada para os bits referentes a conversão ADC.



Desta forma, há a seguinte relação matemática entre o resultado da conversão ADC e a medição da entrada de corrente.



$$\frac{ADC\ -0}{4095\ -0} = \frac{A_{IN} -0}{7,0-0}$$

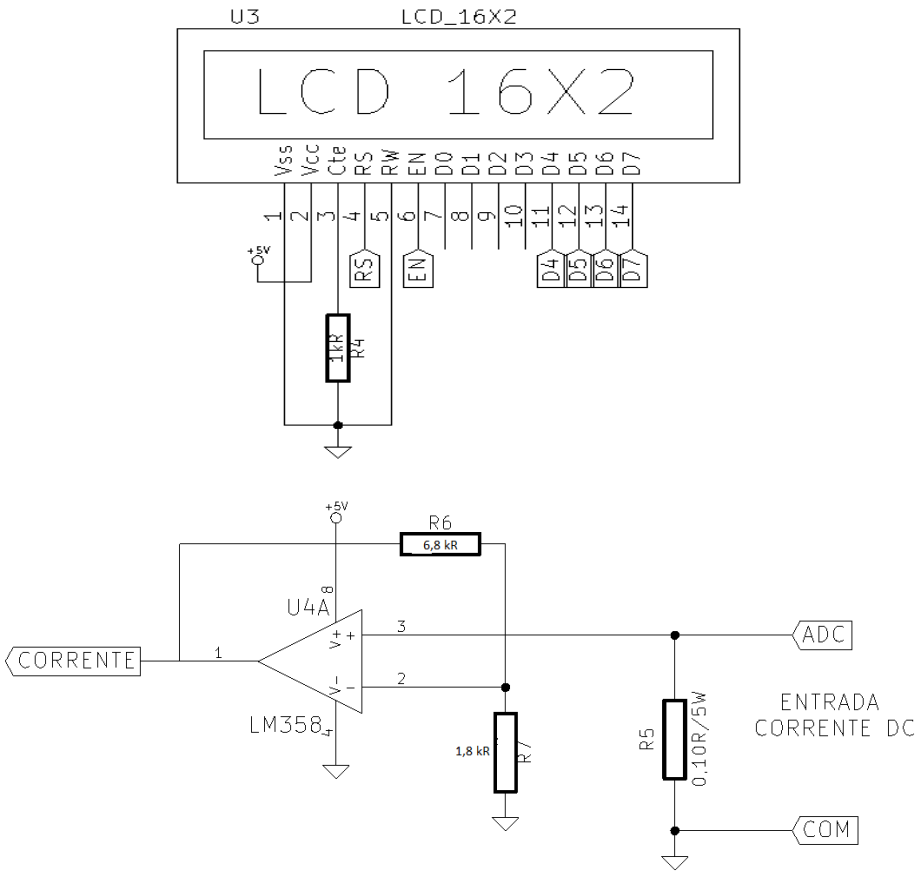
$$\frac{ADC}{4095} = \frac{A_{IN}}{7,0}$$

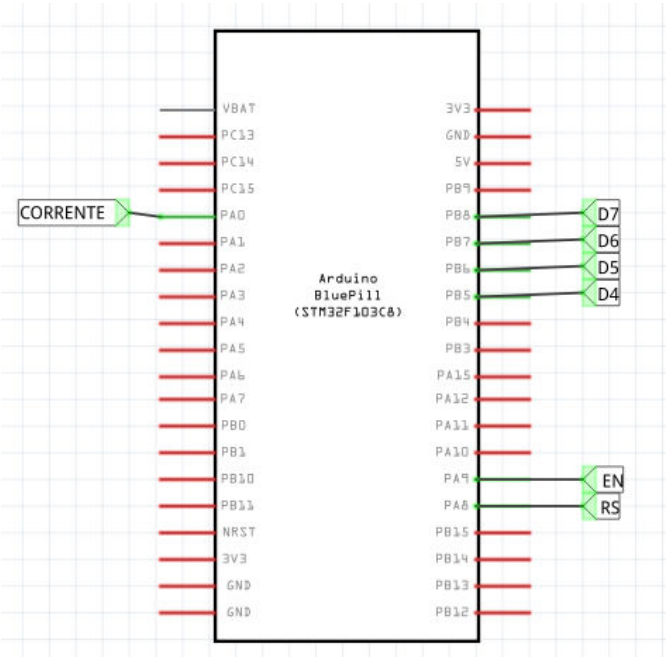
$$A_{IN} = 7,0.\frac{ADC}{4095}$$

Ou seja, após a leitura da entrada analógica, deve-se executar o cálculo matemático acima de forma a obter o valor da corrente de entrada medida pelo circuito e apresentá-la no display.

2. Esquema elétrico

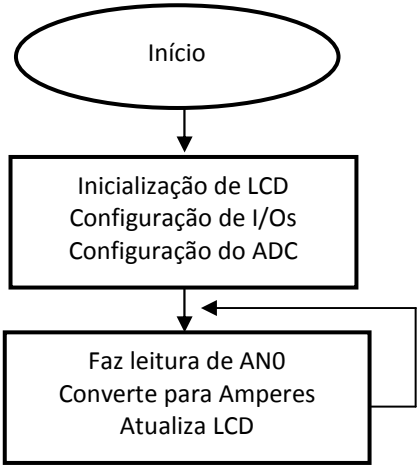
O esquema elétrico para este projeto está apresentado a seguir, onde a entrada de tensão DC está conectada a entrada analógica 0 do ARM. Além disso, o display LCD permite apresentar a corrente medida.





3. Fluxograma

O algoritmo para o desenvolvimento deste projeto está apresentado a seguir.



4. Código fonte

O código fonte completo seguindo o algoritmo visto anteriormente está apresentado a seguir. Observe que no loop principal o programa fica constantemente lendo a entrada analógica AN0 para que em seguida aplique a fórmula de conversão de ADC para corrente de entrada, atualizando o LCD em seguida.

```
void lcd_inic(void);
void lcd_locate(char linha,char coluna);
void lcd_inst4(unsigned char dado);
void lcd_inst(unsigned char dado);
void lcd_dado(char dado);
void lcd_texto(char linha, char coluna, char *texto);

#define  RS     GPIO_PIN_8
#define  EN     GPIO_PIN_9
#define  D4     GPIO_PIN_5
#define  D5     GPIO_PIN_6
#define  D6     GPIO_PIN_7
#define  D7     GPIO_PIN_8

ADC_HandleTypeDef hadc1;
uint16_t AD_RES = 0;
char texto[20];

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_ADC1_Init();
  HAL_ADCEx_Calibration_Start(&hadc1);

  lcd_inic();

  while (1)
  {
 HAL_ADC_Start(&hadc1);
 HAL_ADC_PollForConversion(&hadc1, 1);
 AD_RES = HAL_ADC_GetValue(&hadc1);

 AD_RES=A(D_RES*7.0)/1023;
 itoa(AD_RES, texto, 10);
 lcd_texto(1,1,texto);

 HAL_Delay(1000);
  }
}

void lcd_texto(char linha, char coluna, char *texto)
{
       lcd_locate(linha,coluna);
 do
 {
        lcd_dado(*texto);
        *texto++;
 }while(*texto);
```

```
}

void lcd_inic(void)
{
   lcd_inst4(0x03);
   HAL_Delay(15);
   lcd_inst4(0x03);
   HAL_Delay(15);
   lcd_inst4(0x03);
   HAL_Delay(15);
   lcd_inst4(0x02);
   HAL_Delay(15);

   lcd_inst(0x28);
   HAL_Delay(15);
   lcd_inst(0x0C);
   HAL_Delay(15);
   lcd_inst(0x06);
   HAL_Delay(15);
   lcd_inst(0x01);
   HAL_Delay(100);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *             Rotina de posicionamento de LCD           *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void lcd_locate(char linha,char coluna)
{
   if (linha==1)
      linha=0x80 + --coluna;
   else
      linha=0xC0 + --coluna;

   lcd_inst(linha);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *           Rotina de envio de comando para LCD         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void lcd_inst4(unsigned char dado)
{
   HAL_GPIO_WritePin(GPIOA,RS,GPIO_PIN_RESET);
   HAL_Delay(1);

   if (dado & 0x08)
     HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_SET);
   else
     HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_RESET);
```

```
   if (dado & 0x04)
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_RESET);

   if (dado & 0x02)
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_RESET);

   if (dado & 0x01)
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_RESET);


   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_SET);
   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_RESET);
   HAL_Delay(5);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *           Rotina de envio de comando para LCD       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void lcd_inst(unsigned char dado)
{
   HAL_GPIO_WritePin(GPIOA,RS,GPIO_PIN_RESET);
   HAL_Delay(1);

   if (dado & 0x80)
     HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_SET);
   else
     HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_RESET);

   if (dado & 0x40)
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_RESET);

   if (dado & 0x20)
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_RESET);

   if (dado & 0x10)
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_SET);
```

```
   else
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_RESET);

   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_SET);
   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_RESET);

   if (dado & 0x08)
    HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_SET);
   else
    HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_RESET);

   if (dado & 0x04)
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_RESET);

   if (dado & 0x02)
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_RESET);

   if (dado & 0x01)
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_SET);
   else
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_RESET);

   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_SET);
   HAL_Delay(1);
   HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_RESET);
   HAL_Delay(5);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *          Rotina de envio de dados para LCD         *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void lcd_dado(char dado)
{
   HAL_GPIO_WritePin(GPIOA,RS,GPIO_PIN_SET);
   HAL_Delay(1);

   if (dado & 0x80)
    HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_SET);
   else
    HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_RESET);

   if (dado & 0x40)
```

```
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_RESET);

    if (dado & 0x20)
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_RESET);

    if (dado & 0x10)
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_RESET);

    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_SET);
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_RESET);

    if (dado & 0x08)
      HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_SET);
    else
      HAL_GPIO_WritePin(GPIOA,D7,GPIO_PIN_RESET);

    if (dado & 0x04)
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D6,GPIO_PIN_RESET);

    if (dado & 0x02)
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D5,GPIO_PIN_RESET);

    if (dado & 0x01)
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_SET);
    else
            HAL_GPIO_WritePin(GPIOA,D4,GPIO_PIN_RESET);

    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_SET);
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOA,EN,GPIO_PIN_RESET);
    HAL_Delay(5);
}
```

Capítulo 3

Motor de passo bipolar

1. Visão geral sobre motor de passo

Um motor de passo é utilizado quando precisa-se posicionar com precisão um dispositivo, onde com o uso de um motor CC ficaria impraticável tal controle. Diferente de um motor CC que é composto de dois fios, normalmente um motor de passo é composto de 6 ou 5 fios se for do tipo unipolar e 4 se for bipolar. No motor de passo unipolar, há normalmente 4 bobinas em que acionando as mesmas de maneira correta, é verificado que esta anda o passo determinado pelo fabricante do motor. Observe a figura apresentada abaixo a fim de compreender melhor a formação de um motor de passo.

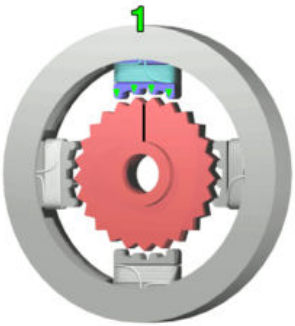


Figura 1: Aspecto típico de um motor de passo bipolar
Fonte: pt.wikipedia.org

No motor de passo unipolar há internamente a composição de 4 bobinas, onde acionando uma de cada vez em sequência é observado que o motor fornece um giro completo. Por exemplo, suponha que por passo este motor ande 1,8º. Sendo assim, para que ele dê uma volta completa, é necessário acionar o motor de passo a seguinte quantidade de passos considerando 360º uma volta completa.

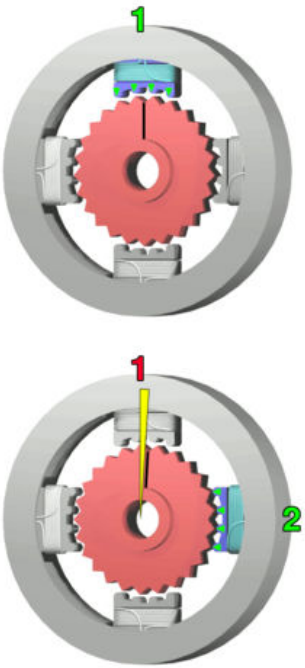
$$360º/\ 1{,}8º = 200\ \text{Passos}$$

Um dos principais fatores para aquisição de um motor de passo é saber quantos graus por passo ele possui. No caso acima, o motor de passo anda 1,8º por passo, porém existem motores de passos mais precisos em que tal faixa pode diminuir bastante assim como os não muito precisos, em que pode ser ainda maior a quantidade de graus por passo. Suponha que a precisão de um motor de passo é de 7,2º por passo. Quantos passos este motor tem que dar para andar 360º? Basta novamente repetir o cálculo acima, como mostra o cálculo abaixo:

$$360º/\ 7{,}2º\ º = 50\ \text{Passos}$$

Atente quando for comprar um motor de passo, para adquirir um que atenda sua necessidade no projeto.

A seguir o ciclo completo para que o motor de passo ande 3,6º considerando que ele tem a precisão de 1,8º por passo:



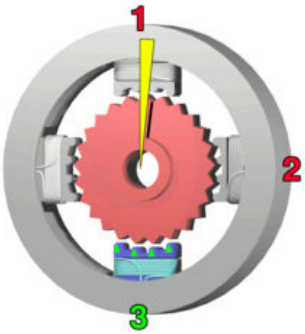


Figura 2: Passo para andar 3,6º com o motor de passo
Fonte: pt.wikipedia.org

Na primeira figura o motor de passo está desativado. Em seguida, a bobina 2 é acionada e faz andar 1,8º. Para completar 3,6º, a bobina 2 é desativada, porém a 3 fica ativa o que faz que ande mais 1,8º resultando ao todo em 3,6º. Tal conceito permanece ao dar uma volta completa ou andar mais alguns passos com o motor de passo. Caso o programa voltasse os 3,6º a sequência gerada seria a seguinte.

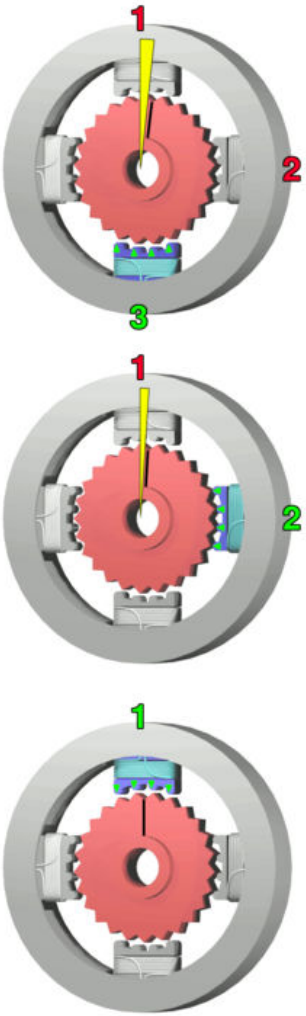


Figura 3: Passo para voltar 3,6º com o motor de passo
Fonte: pt.wikipedia.org

Foi gerada a sequência na lógica inversa para que o motor possa assim completar o ciclo desejado. Para acionar o motor de passo em um sentido ou outro basta que a sequência aplicada ao motor se inverta.

No mercado de componentes normalmente encontra-se três tipos de motores de passo, a saber: Relutância Variável, Imã Permanente e Híbrido. O primeiro e mais comum é o de Relutância variável, onde neste tipo de motor o estator é formado por quatro polos. A sequência de rotação deste motor depende exclusivamente da sequência de energização feita na bobina. Quando as bobinas ficam desenergizadas, o rotor fica solto podendo assim mover o motor externamente.

Já o motor do tipo imã permanente apresenta sua estrutura muito parecida com a relutância variável, porém ele fica parado caso o motor deixe de ficar energizado em função do próprio imã permanente. O motor do tipo híbrido usa características dos dois últimos motores apresentados.

Tais motores são hoje utilizados em uma infinidade de aplicações, a saber: Drivers de Disquetes, Impressoras, Scanners, Injeção Eletrônica, Posicionamento de antenas dentre outros.

Na compra de um motor de passo, deve-se atentar a alguns fatores importantes, conforme listado a seguir.

Graus por Passo = É a característica que determina quantos graus o motor andará por passo. Valores típicos são: 0,72º,1,8º, 3,6º graus por passo.

Momento de Frenagem = É o momento máximo com o rotor bloqueado, sem perda de passos.

Momento (Torque) = Referente ao torque que o motor terá.

Resistência Ôhmica = Referente ao consumo que o motor possui caso uma bobina fique ativa.

Holding Torque = é mínima potência para fazer o motor mudar de posição parada.

Resposta de Passo = Tempo no qual o motor de passo precisa para dar um passo.

Ressonância = Referente à frequência de oscilação do motor que determina a frequência em que caso o mesmo venha a funcionar poderá oscilar e fazê-lo perder passo.

Tensão de trabalho = É referente à tensão no qual o motor de passo funciona. Tensões típicas variam de 3V a 48V.

2. Motor de passo bipolar

Tipicamente este motor é encontrado no mercado com 4 fios, como sugere a próxima figura.

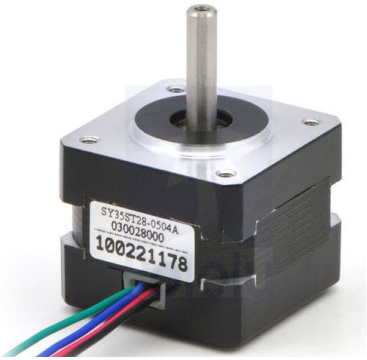


Figura 4: Aspecto típico de um motor de passo bipolar
Fonte: http://multilogica-shop.com/motor-de-passo-bipolar-200-passosvolta-35x28mm-10v-500ma

Informações importantes a serem obtidas no ato da aquisição do motor são: tamanho, peso, diâmetro do eixo, passos por revolução, consumo de corrente por fase, voltagem de operação, resistência por enrolamento e torque. As especificações do motor apresentado, cujo modelo é SY35ST28-0504A estão listadas a seguir.

Especificações	
Tamanho	35mmx35mmx28mm
Peso	140g
Diâmetro do eixo	5mm
Passos por revolução	200
Resolução(o/passo)	1,8
Consumo de corrente por fase	500mA
Voltagem de operação	10V
Resistência por fase	20Ω
Torque	1000 g.cm

Tabela 1: Especificações do motor de passo bipolar
Fonte: http://multilogica-shop.com/motor-de-passo-bipolar-200-passosvolta-35x28mm-10v-500ma

Internamente este motor é composto de duas bobinas independentes com um rotor magnético, como sugere a figura a seguir.

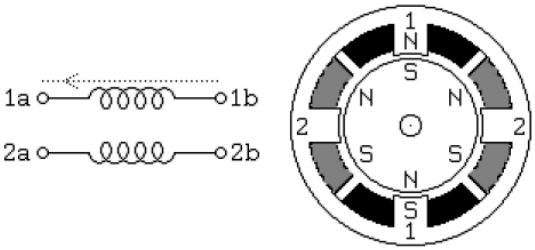


Figura 5: Configuração das bobinas
Fonte: alfa.facens.br/~sidney/controle_motor_passo.pdf

Quando uma bobina é acionada o eixo magnético é atraído por esta, levando assim ao movimento, ou seja, o passo do motor.

Observe que o polo é gerado de acordo com o sentido da corrente da bobina, sendo necessário controlar qual polo fica positivo

de acordo com o passo do motor. Para isso, faz-se uso de uma ponte H que possibilita controlar a polaridade do polo magnético.

3. Driver L298

A próxima figura mostra o esquema típico de uma ponte H com transistores.

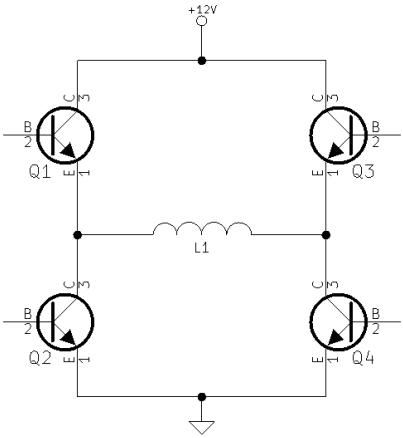


Figura 6: Ponte H

Quando Q1 e Q4 ficam habilitados a bobina é alimentada com o sentido mostrado na próxima figura.

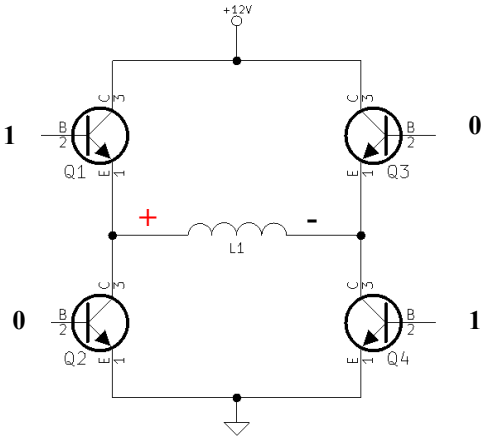


Figura 7: Ponte H alimentando bobina em sentido direto

Quando Q2 e Q3 estão habilitados a bobina é alimentada com o sentido mostrado na próxima figura.

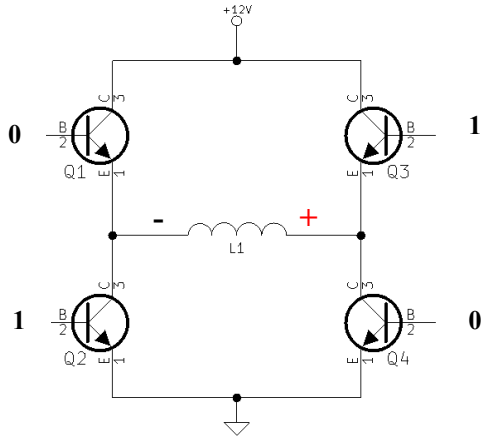


Figura 8: Ponte H alimentando bobina em sentido inverso

É necessário utilizar dois conjuntos de ponte H para controlar as duas bobinas do motor de passo bipolar. Entretanto, há disponível no mercado eletrônico diversas soluções encapsuladas em um único circuito integrado que fornecem duas pontes H integradas em um só encapsulamento, facilitando assim o controle deste tipo de projeto. Um destes drivers é o L298 que é um *dual full-bridge driver,* conforme ilustra a figura a seguir.

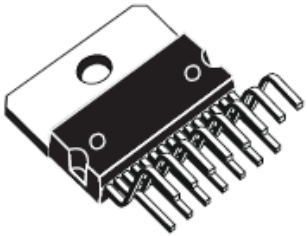


Figura 9: Encapsulamento do L298
Fonte: L298 Datasheet

As características deste driver estão listadas na tabela abaixo.

Características máximas	
Alimentação Vs (conectada ao motor)	50V
Alimentação Vss (lógica de funcionamento)	7V
Corrente máxima em operação contínua	2A
Dissipação de potência máxima	25 W

Tabela 2: Especificações do L298
Fonte: L298 Datasheet

Internamente ele é construído de acordo com a figura apresentada abaixo, onde verifica-se dois drivers de ponte H.

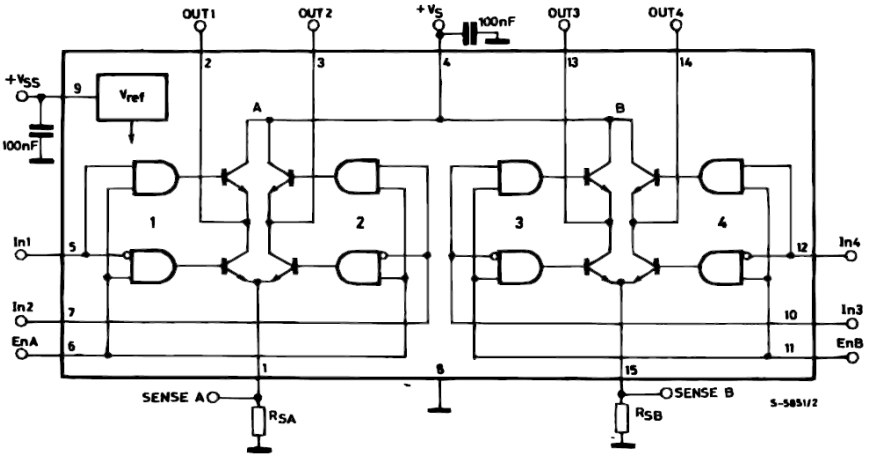


Figura 10: Construção interna do L298
Fonte: L298 Datasheet

Cada ponte H é controlada individualmente. Por exemplo, os pinos In1, In2 e EnA controlam o driver a esquerda enquanto que os pinos In3, In4 e EnB o driver a direita. Adicionalmente, há em cada driver a possibilidade de conectar um resistor do tipo shunt para medir a corrente que passa pela bobina e desta forma evitar o aquecimento e queima tanto do driver quanto da bobina.

A próxima figura mostra a localização dos pinos no driver L298.

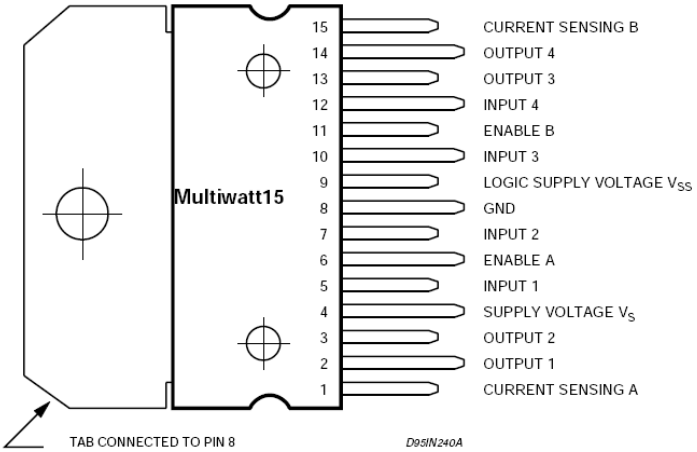


Figura 11: Localização dos pinos do L298
Fonte: L298 Datasheet

A tabela verdade que relaciona o funcionamento dos pinos In1, In2 e EnA estão apresentados na próxima tabelar. Esta mesma lógica é válida para o outro driver disponível no L298.

EnA	In1	In2	Out1	Out2
0	X	X	NC	NC
1	1	0	Vs	GND
1	0	1	GND	Vs

*NC= Não conectado
Tabela 3: Níveis lógicos para controle do L298
Fonte: L298 Datasheet

Para garantir que a corrente consumida pelo motor de passo não ultrapasse o máximo suportado que é de 500 mA ou 0,5A, um resistor shunt de 0,10Ω foi colocado em série com GND nas entradas

SENSE_A e SENSE_B como sugere a próxima figura e controla as entradas de habilitação EnA e EnB, onde caso a corrente ultrapasse o limite suportado pelo motor de passo a ponte corta o fornecimento de energia a carga. Este circuito é chamado de *chopper.*

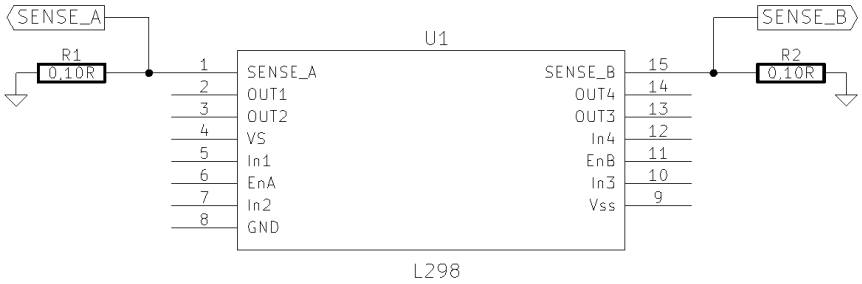


Figura 12: Resistor shunt de 0,10Ω colocado nas entradas SENSE_A e SENSE_B

Quando a corrente for 0,5A a tensão verificada nos pontos SENSE_A ou SENSE_B é determinada pela 1º Lei de Ohm, como demonstrado a seguir.

$$V = R.I$$
$$V = 0{,}10.0{,}5$$
$$V = 0{,}05V$$

Esta tensão é amplificada em 48 através de um amplificador operacional não inversor, como sugere a próxima figura.

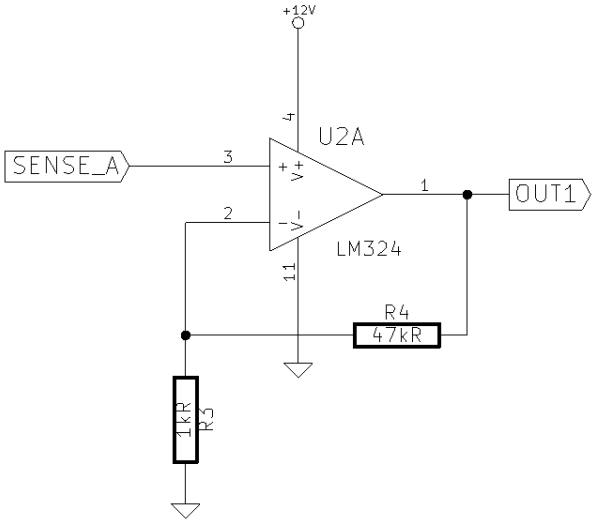


Figura 13: Circuito amplificador para SENSE_A

Na saída OUT1 é observado à tensão de 2,4V que é o resultado do produto de 0,05V por 48 quando a corrente ao passar pela bobina do motor de passo for de 0,5A. Neste instante, o driver deve deixar de fornecer energia ao motor de passo, pois caso contrário pode aquecê-lo demasiadamente comprometendo sua eficiência. Neste instante a entrada EnA ou EnB do driver L298 deve ficar desabilitada, ou seja, em nível 0. Desta forma, a saída OUT1 fica conectada a uma entrada de comparador de tensão, onde enquanto a tensão for menor ou igual a 2,5V a saída de habilitação está em 1, entretanto ao passar deste limiar de tensão a saída fica em nível 0 fazendo que o driver deixe de operar naquele instante, sempre mantendo a

corrente na faixa correta de operação do motor de passo. Acompanhe o circuito completo a seguir.

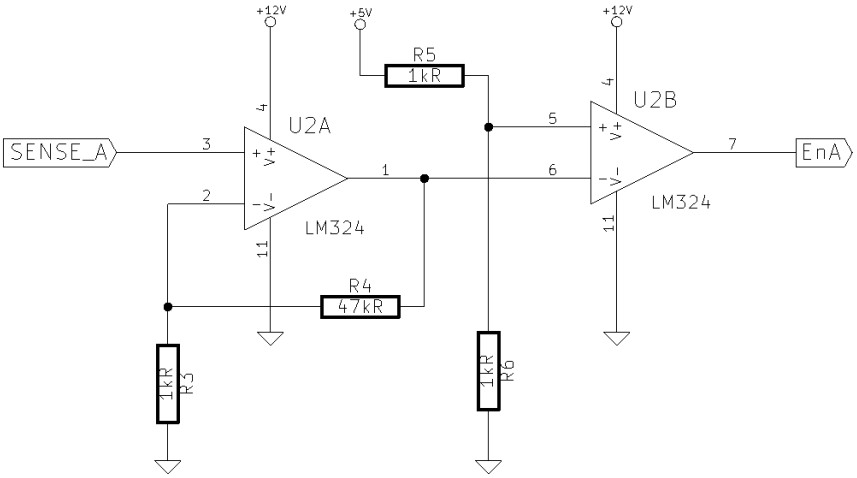
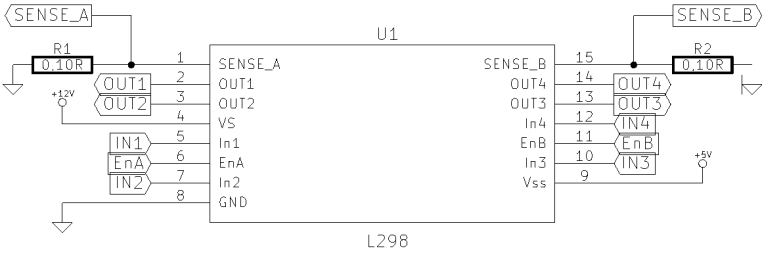


Figura 14: Circuito amplificador e comparador para SENSE_A

Este mesmo circuito está disponível para a outra ponte H do driver L298 referente à SENSE_B.

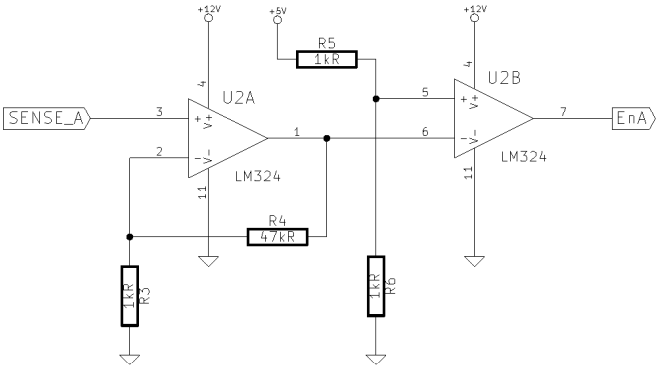
4. Esquema elétrico

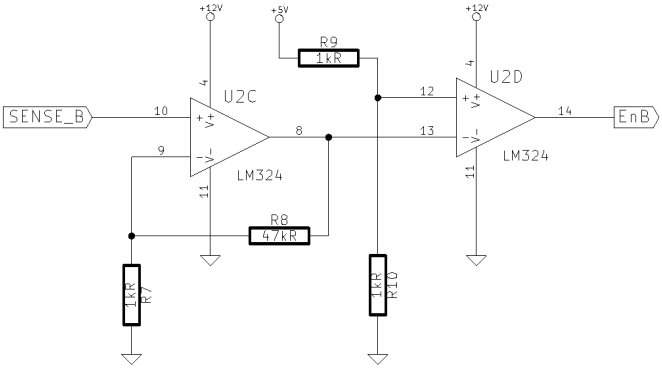
A seguir o esquema elétrico completo a ser utilizado nas experiências a serem demonstradas nos próximos tópicos.





MOTOR DE PASSO BIPOLAR





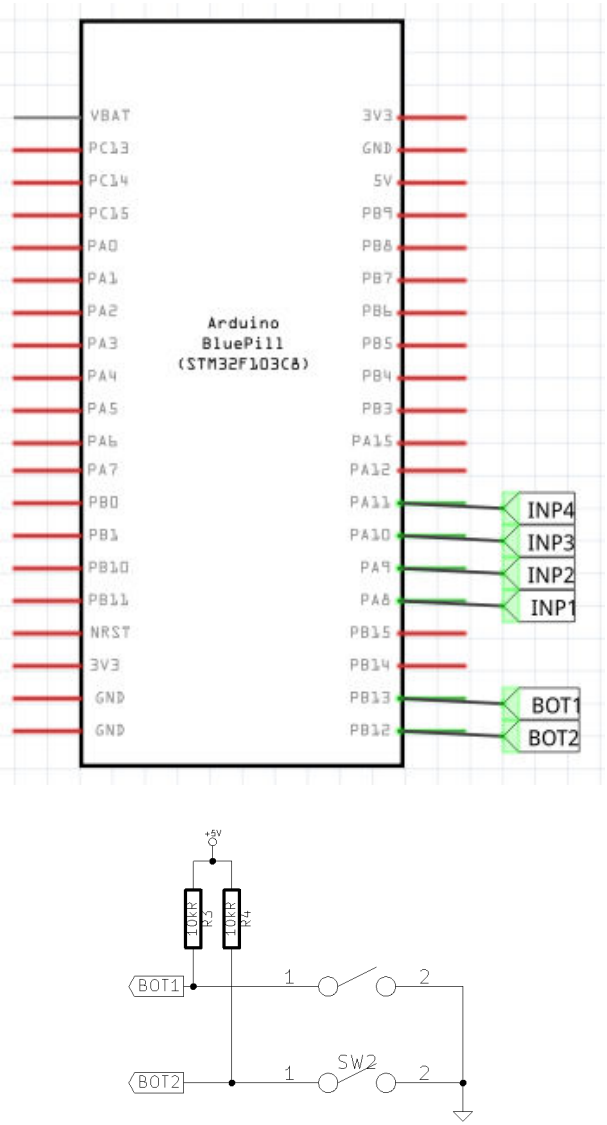


Figura 15: Esquema elétrico do experimento

5. Controle em modo full step

O controle *full step* (passo completo) é aquele no qual o motor de passo move-se o especificado a cada passo de acordo com o datasheet, ou seja, 1,8º, 3,6º, 7,2º, etc. A próxima figura procura mostrar como é feito este acionamento considerando as duas bobinas dispostas internamento ao motor, onde na vertical está representada a bobina 1 e na horizontal a bobina 2.

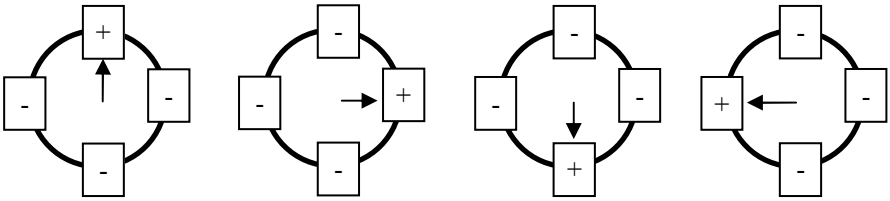


Figura 16: Passos para controle full step

Desta forma a sequência de acionamento das bobinas 1 e 2 ficam conforme apresentado na tabela abaixo.

Sequência de acionamento das bobinas 1 e 2				
Bobina 1	+12V	0V	-12V	0V
Bobina 2	0V	+12V	0V	-12V

Tabela 4: Sequência de acionamento das bobinas

A tensão de +12V é obtida deixando a entrada In1=1 e In2=0 para o driver 1 e In3=1 e In4=0 para o driver 2. A tensão de -12V é alcançada deixando In1=0 e In2=1 para o driver 1 e In3=0 e In4=1

para o driver 2. Quando as entradas In1 e In2 ou In3 e In4 estiverem no mesmo nível, a saída fica em 0V.

Realizando o ciclo acima o motor move 4 passos, ou seja, $7,2^{o}$ já que ele possui precisão de $1,8^{o}$/passo em modo full step.

5.1 Fluxograma

O exemplo demonstrado neste tópico permite mover o motor no sentido horário ou anti-horário de acordo com o botão pressionado nos pinos digitais 0 e 2 conforme o esquema elétrico. Acompanhe na próxima figura o fluxograma deste experimento.

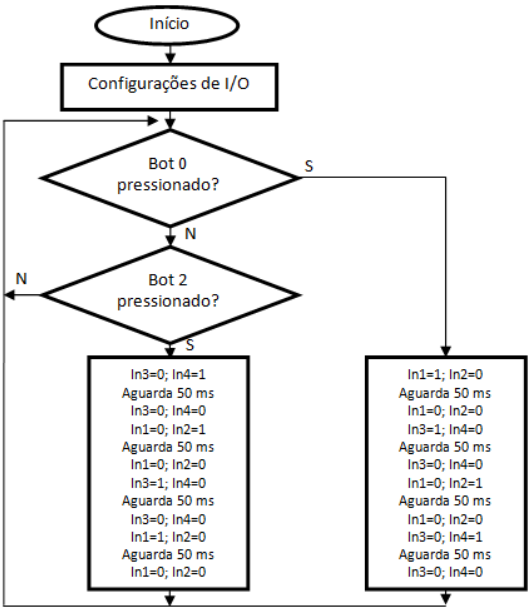


Figura 17: Fluxograma para controle full step

Observe que a diferença entre um bloco e outro está ligada a sequência de acionamento do motor.

5.2 Código fonte

O código fonte que implementa o fluxograma demonstrado anteriormente pode ser visualizado a seguir.

```
#define In1      GPIO_PIN_8
#define In2      GPIO_PIN_9
#define In3      GPIO_PIN_10
#define In4      GPIO_PIN_11
#define Bt1      GPIO_PIN_13
#define Bt2      GPIO_PIN_12

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  while (1)
  {
        if(HAL_GPIO_ReadPin(GPIOA,Bt1)==0)
        {
              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
              HAL_Delay(50);
              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);

              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
              HAL_Delay(50);
              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);

              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
              HAL_Delay(50);
              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);

              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
```

```
                HAL_Delay(50);
                HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
        }

        if(HAL_GPIO_ReadPin(GPIOA,Bt2)==0)
        {
                HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
                HAL_Delay(50);
                HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);

                HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
                HAL_Delay(50);
                HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);

                HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
                HAL_Delay(50);
                HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);

                HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
                HAL_Delay(50);
                HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        }
  }

}
```

6. Controle em modo half step

Neste modo de operação o motor de passo é capaz de mover metade do passo especificado, ou seja, se o motor de passo move a 1,8°/passo o mesmo será capaz de mover-se a 0,9°/passo. Isso garante maior precisão ao motor, no entanto, faz com que ele tenha um maior aquecimento já que para isso é necessário utilizar as duas

bobinas de forma simultânea diferente do modo full step que apenas uma bobina fica ativa por vez. Além disso, a velocidade para completar uma volta no motor é menor se comparado ao modo full step.

A próxima figura mostra como é feito este acionamento considerando as duas bobinas dispostas internamento ao motor, onde na vertical está representada a bobina 1 e na horizontal a bobina 2.

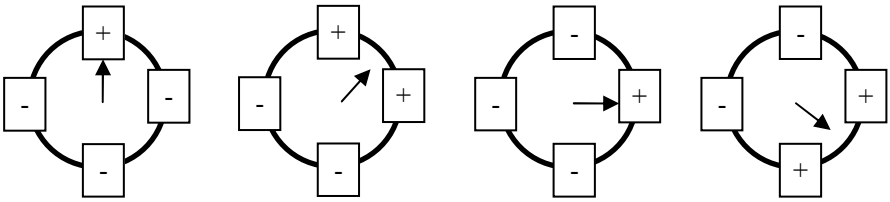


Figura 18: Passos para controle half step

Desta forma a sequência de acionamento das bobinas 1 e 2 ficam conforme apresentado na tabela abaixo.

Sequência de acionamento das bobinas 1 e 2								
Bobina 1	+12V	+12V	0V	-12V	-12V	-12V	0V	+12V
Bobina 2	0V	+12V	+12V	+12V	0V	-12V	-12V	-12V

Tabela 5: Sequência de acionamento das bobinas

Realizando o ciclo acima o motor move 4 passos, ou seja, $3,6^{o}$ já que ele possui precisão de $1,8^{o}$/passo em full step e $0,9^{o}$/passo em half step.

6.1 Fluxograma

O exemplo demonstrado neste tópico permite mover o motor no sentido horário ou anti-horário de acordo com o botão pressionado nos pinos digitais 0 e 2 conforme o esquema elétrico em modo half step. Acompanhe na próxima figura o fluxograma deste experimento.

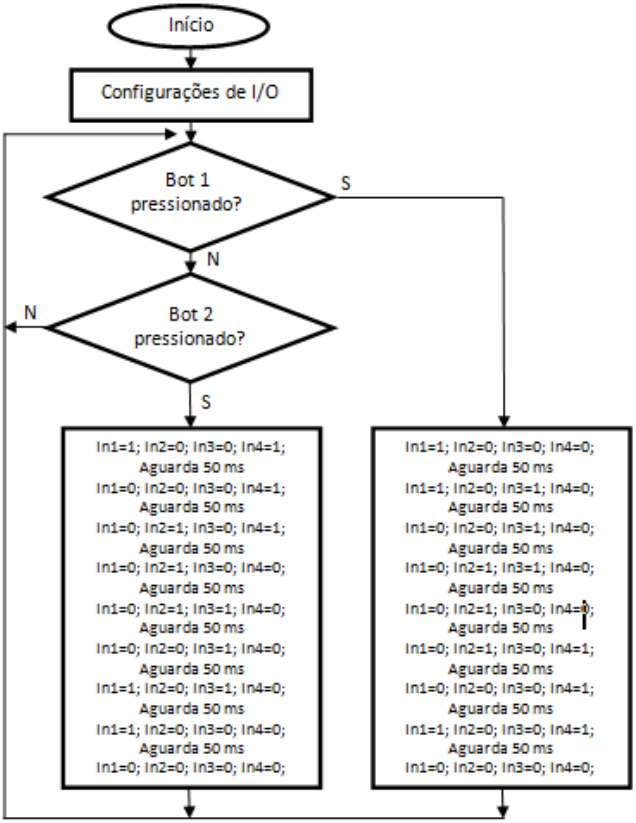


Figura 19: Fluxograma para controle half step

A diferença principal entre um bloco e outro está ligada a sequência de acionamento do motor que no botão 2 está invertida em relação ao botão 1.

6.2 Código fonte

O código fonte que implementa o fluxograma demonstrado anteriormente pode ser visualizado a seguir.

```
#define In1      GPIO_PIN_8
#define In2      GPIO_PIN_9
#define In3      GPIO_PIN_10
#define In4      GPIO_PIN_11
#define Bt1      GPIO_PIN_13
#define Bt2      GPIO_PIN_12

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();

  while (1)
  {
        if(HAL_GPIO_ReadPin(GPIOA,Bt1)==0)
        {
              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
              HAL_Delay(50);

              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
              HAL_Delay(50);

              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
              HAL_Delay(50);

              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_SET);
              HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
              HAL_Delay(50);

              HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
              HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
```

```
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_RESET);
}


if(HAL_GPIO_ReadPin(GPIOA,Bt2)==0)
{
        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In2,GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA,In3,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA,In4,GPIO_PIN_SET);
        HAL_Delay(50);

        HAL_GPIO_WritePin(GPIOA,In1,GPIO_PIN_RESET);
```